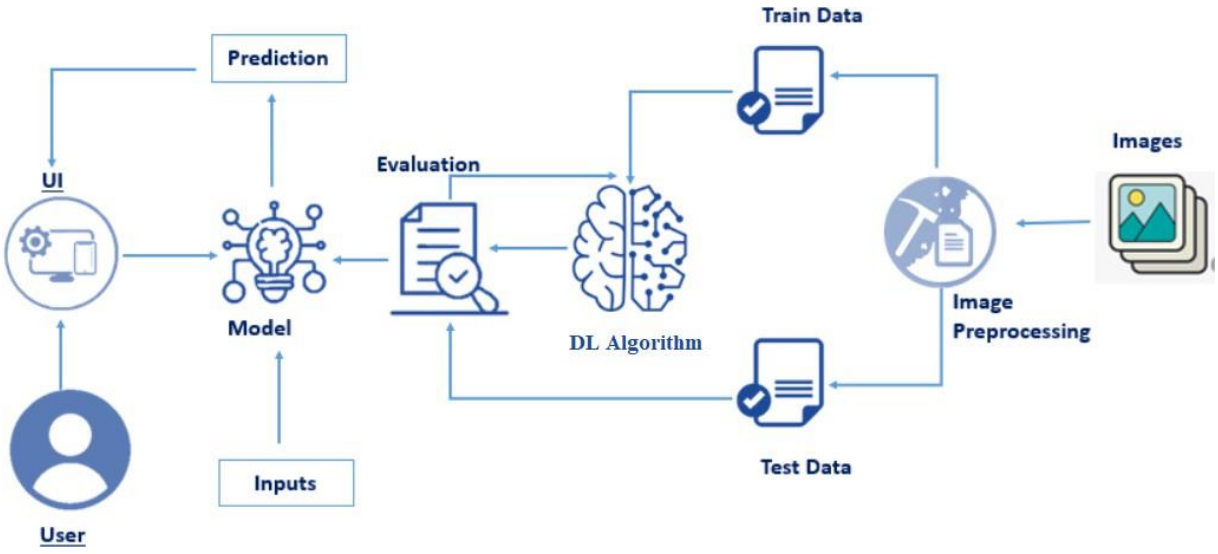
Intelligent Garbage Classification using Deep learning

Introduction:

In today's fast-paced world, the automotive industry is continuously evolving, offering a multitude of vehicle options with varying features, designs, and price points. For consumers, purchasing a car is a significant decision that involves a substantial financial commitment. It is essential to make an informed choice that aligns with one's preferences, needs, and budget. However, the abundance of choices can make this process overwhelming. This is where predictive analytics and machine learning can play a pivotal role. The Car Purchase Prediction Project aims to simplify the car buying process by leveraging data-driven insights to assist potential buyers in making informed decisions. By analyzing historical data on various car models, customer preferences, and market trends, this project seeks to predict the most suitable car options for individual customers, enhancing their overall buying experience. Whether you're a first-time car buyer or looking to upgrade your existing vehicle, this project will empower you to explore your options with confidence and ease.

Technical Architecture:



Prerequisites:

To complete this project, you must require the following software's, concepts, and packages

Google Colab, short for Google Colaboratory, is a cloud-based, free-to-use platform provided by Google for individuals and organizations to write and execute Python code in a collaborative environment. It is particularly popular among data scientists, machine learning practitioners, and researchers for its ease of use and access to powerful computing resources.

1. **To build Machine learning models you must require the following packages**

- **Numpy**:
  - It is an open-source numerical Python library. It contains a multidimensional array and matrix data structures and can be used to perform mathematical operations

Project Objectives:

By the end of this project you will:

- Know fundamental concepts and techniques of AI and ML algorithms
- Know how to pre-process/clean the data using different data preprocessing techniques.

Project Flow:

- The data is analyzed by the model which is integrated with flask application.
- The trained Model analyze the data, then prediction is showcased on the Flask UI.

To accomplish this, we have to complete all the activities and tasks listed below

- Data Collection.
  - Create Train and Test Folders.
- Data Preprocessing.
  - Import the necessary libraries
  - Import the dataset
  - Check for Null values
  - Data Visualization
  - Outlier detection
  - Splitting Dependent and independent features
  - Encoding categorical features
  - Splitting data into Train and Test
- Model Building
  - Import the model building Libraries
  - Initializing the model
  - Training and Testing the model
  - Evaluation of model

Milestone 1: Data Collection

Collect data about cars and their features In this project, we have collected data such as UserId,Gender,EstimatedSalary,Age,Purchased of about 400 people

Downloadt Dataset- https://github.com/ROHIT290803/dataset/blob/main/Car_purchase.csv

Milestone 2: data Preprocessing

Activity 1:Import the Necessary library

Customer Car Purchase Prediction by watching Advertisement on Social Media

Here we used Car_Purchase dataset which providing information regarding the person's age and estimated salary & if he/she is interested in buying a car .(yes=1,No=0) we will predict that what are the chances of new person of some age to be interested in buying car

IMPORTING NECESSARY LIBRARIES

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
```

Activity 2: Import the dataset

IMPORTING DATASET

```
# Importing the dataset
df = pd.read_csv('Car_purchase.csv')
```

Activity 3: Check for Null values

CHECKING FOR NULL VALUES

```
df.isnull().any
```

```
<bound method NDFrame._add_numeric_operations.<locals>.any of      User ID  Gender    Age  EstimatedSalary  Purchased
0      False   False  False            False      False
1      False   False  False            False      False
2      False   False  False            False      False
3      False   False  False            False      False
4      False   False  False            False      False
..       ...     ...    ...              ...        ...
395    False   False  False            False      False
396    False   False  False            False      False
397    False   False  False            False      False
398    False   False  False            False      False
399    False   False  False            False      False

[400 rows x 5 columns]>
```

```
df.isnull().sum()
```

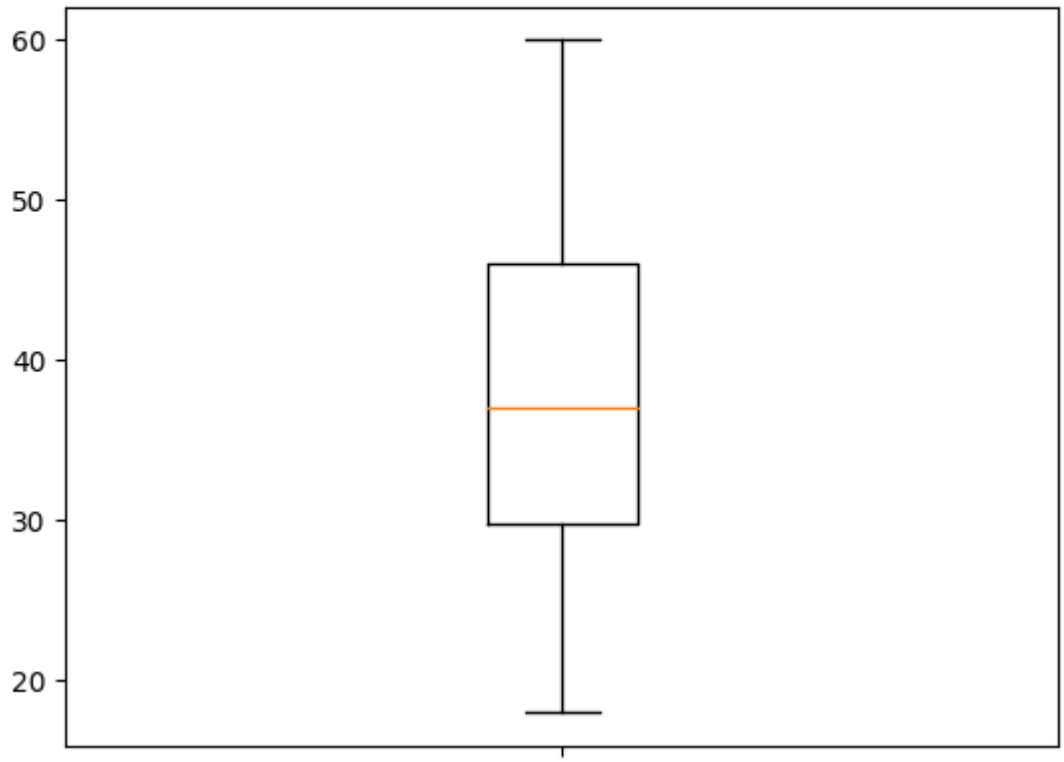
```
User ID            0
Gender             0
Age                0
EstimatedSalary    0
Purchased          0
dtype: int64
```

As we can see there are no null values

Activity 4: Data Visualization

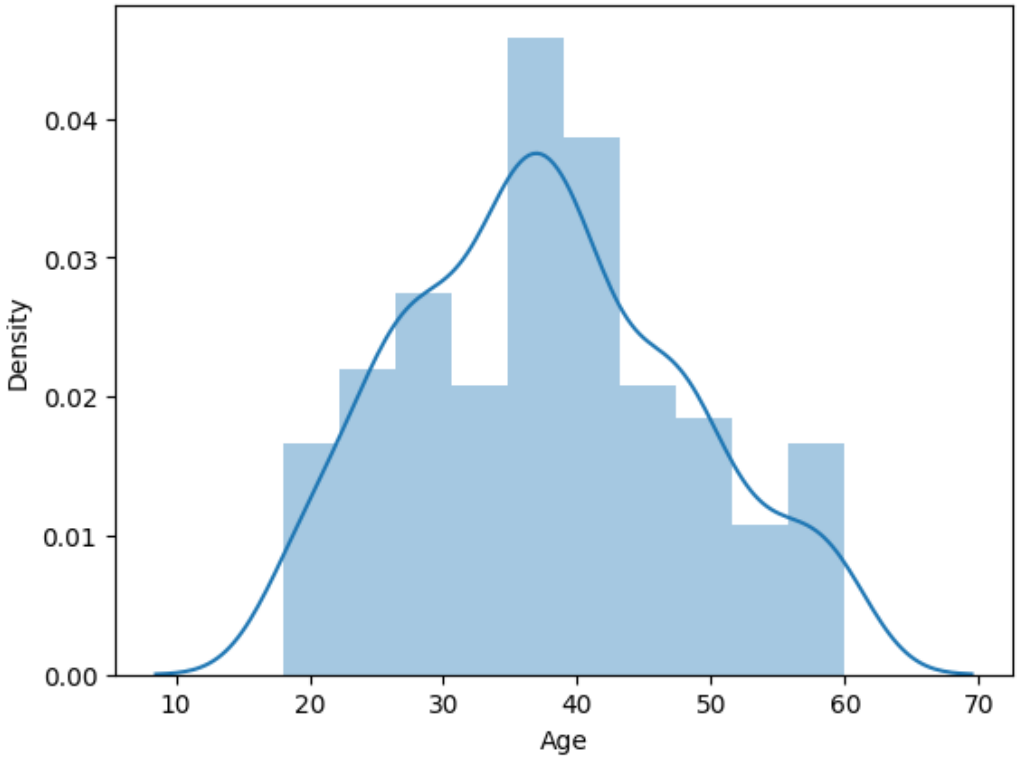
UNIVARIATE ANALYSIS

```
q = list(df.Age)
plt.boxplot(q)
plt.show()
```

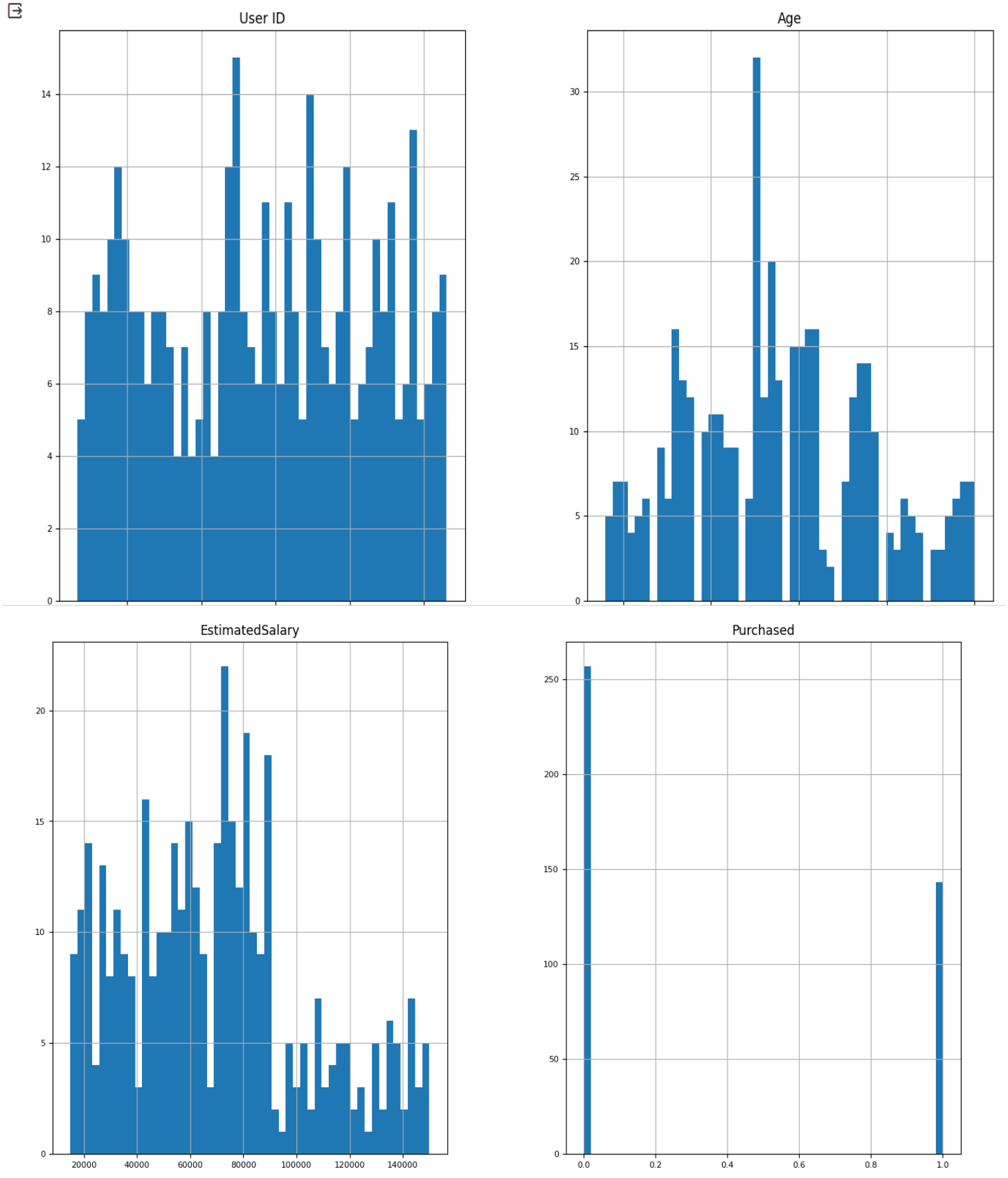


```
sns.distplot(df["Age"])
```

```
<ipython-input-11-cf0334540b62>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df["Age"])
<Axes: xlabel='Age', ylabel='Density'>
```



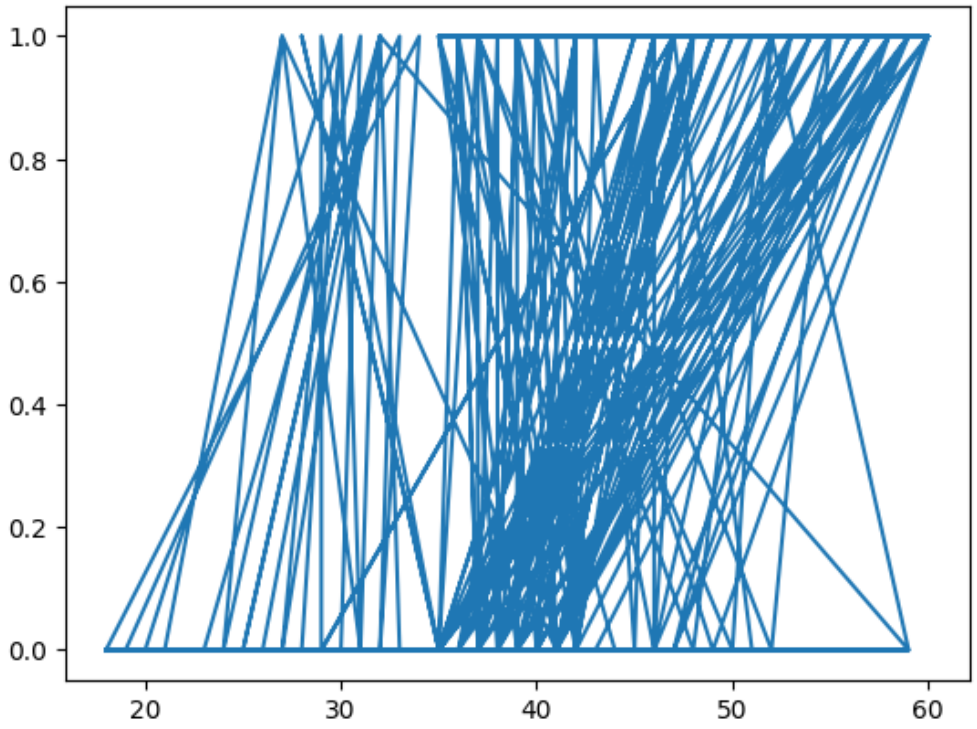
```
df.hist(figsize=(16, 20), bins=50, xlabelsize=8, ylabelsize=8); # ; avoid having the matplotlib verbose informations
```



Bivariate analysis

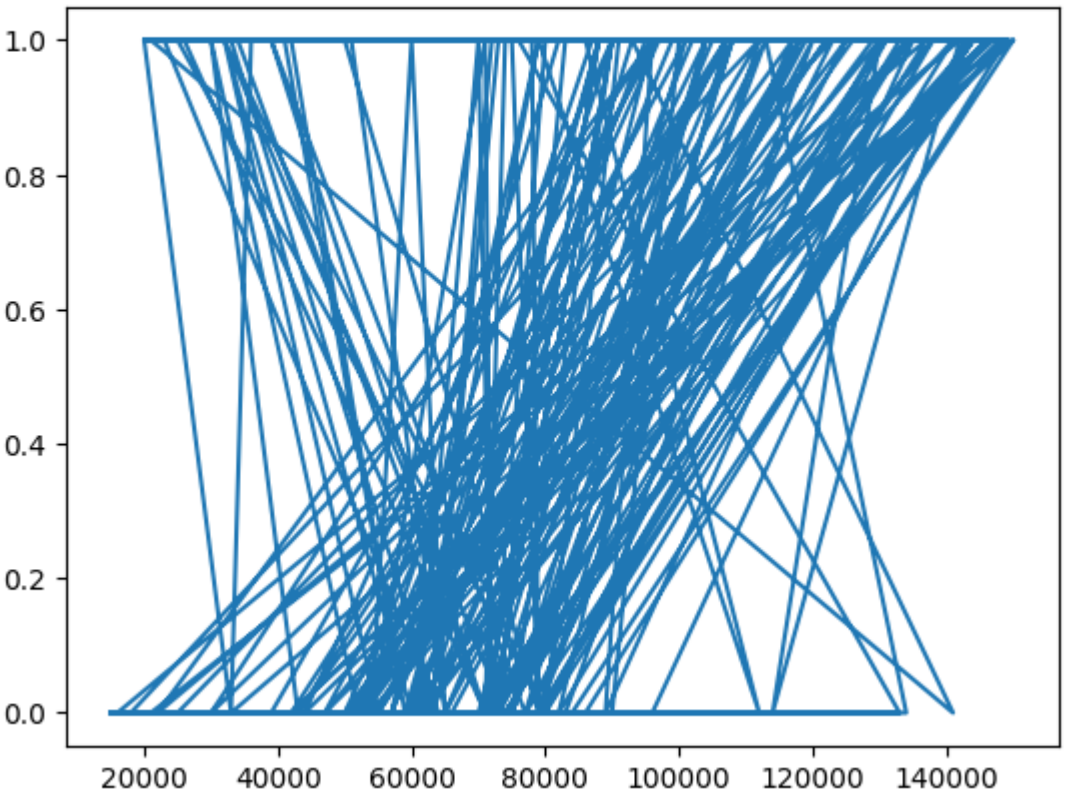
```
plt.plot(df.Age,df.Purchased)
```

```
[<matplotlib.lines.Line2D at 0x7b47c934a710>]
```



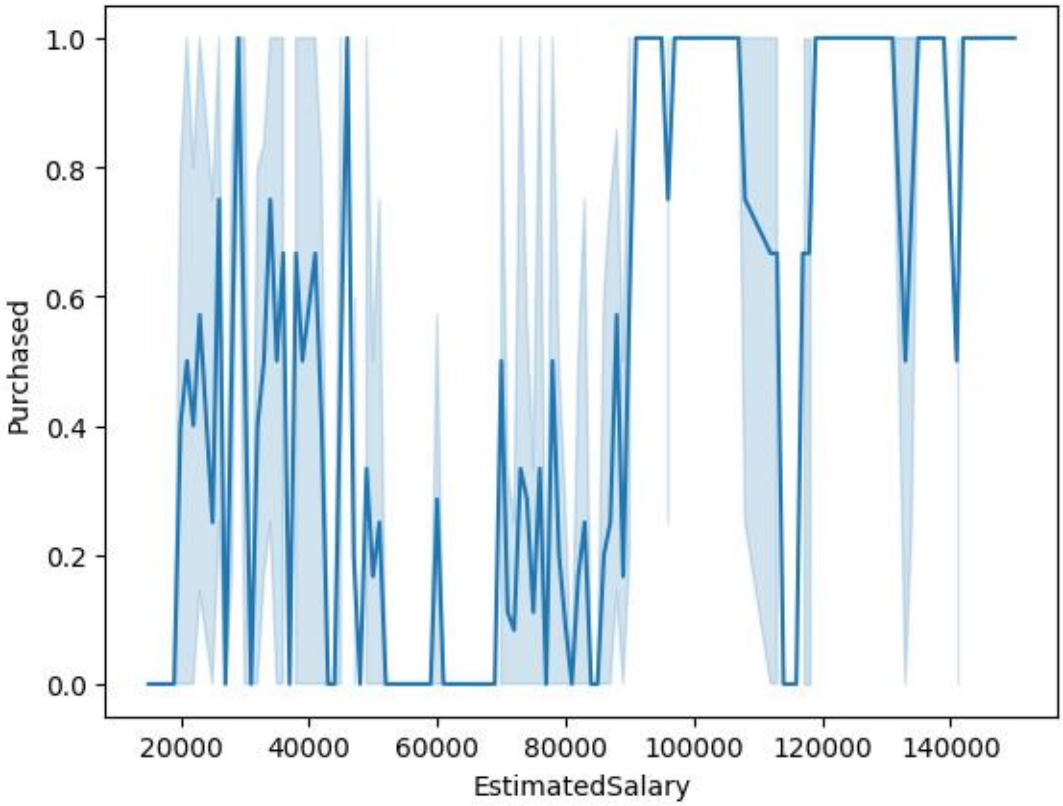

```
plt.plot(df.EstimatedSalary,df.Purchased)
```

```
[<matplotlib.lines.Line2D at 0x7b47c91afe20>]
```



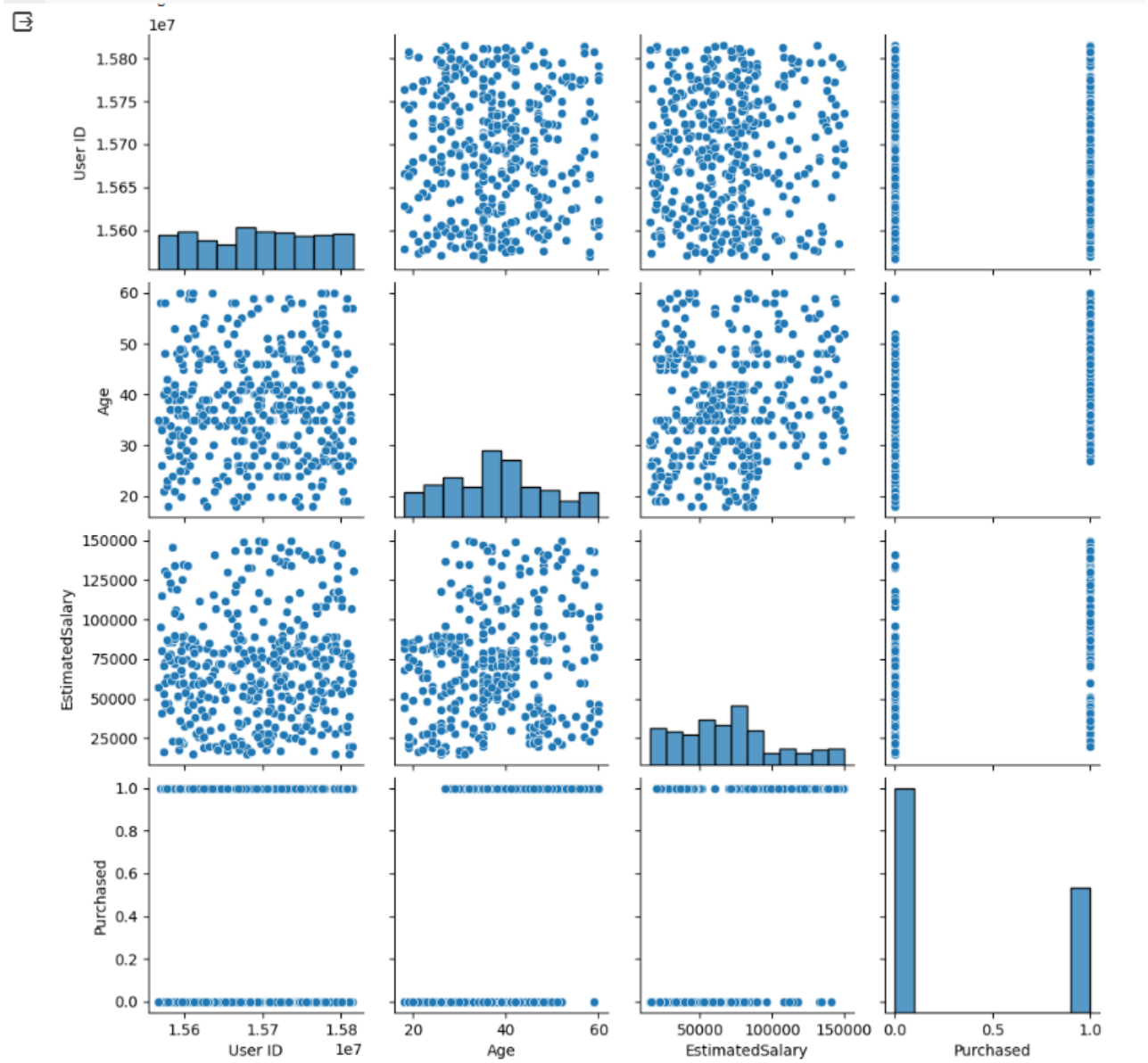
```
sns.lineplot(x="EstimatedSalary",y="Purchased",data=df)
```

```
<Axes: xlabel='EstimatedSalary', ylabel='Purchased'>
```



MULTIVARIATE ANALYSIS

```
sns.pairplot(df)
```

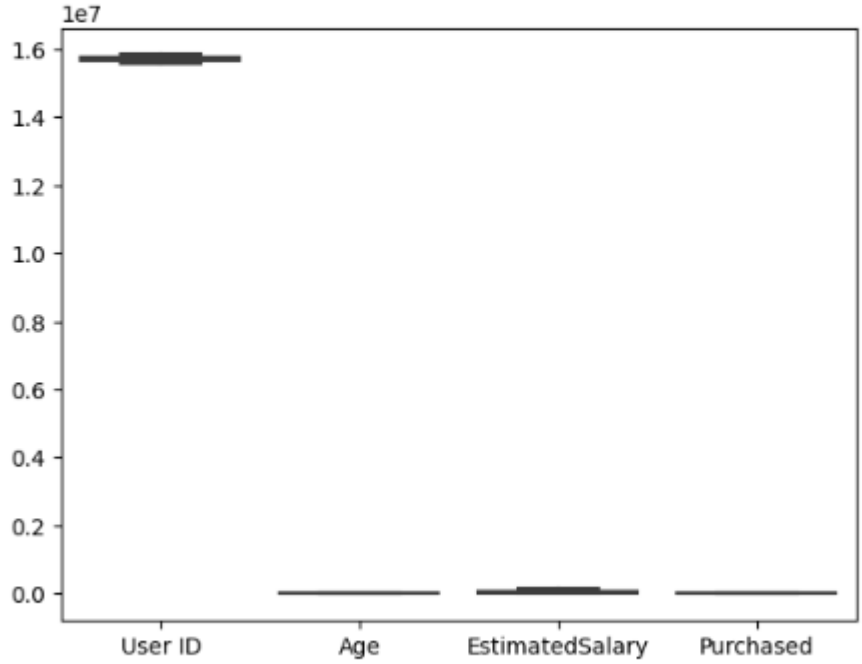


Activity 5: Outlier detection

OUTLIER DETECTION

```
sns.boxplot(df)
```

```
<Axes: >
```



From the figure we come to know that there are no outliers.Hence,they need not be treated

Activity 6: Splitting Dependent and independent features

Splitting the Dataset into dependent and independent variables

```
x=df.iloc[:,1:4]
x.head()
```

	Gender	Age	EstimatedSalary
0	Male	19	19000
1	Male	35	20000
2	Female	26	43000
3	Female	27	57000
4	Male	19	76000

```
y=df.Purchased
y.head()
```

```
0    0
1    0
2    0
3    0
4    0
Name: Purchased, dtype: int64
```

Activity 7: Encoding categorical features

ENCODING THE CATEGORICAL DATA USING LABELENCODER()

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

```
x["Gender"]=le.fit_transform(x["Gender"])
x.head()
```

	Gender	Age	EstimatedSalary
0	1	19	19000
1	1	35	20000
2	0	26	43000
3	0	27	57000
4	1	19	76000

Thus the gender feature has been encoded

Activity 8: Splitting data into Train and Test

SPLITTING THE DATASET INTO TRAINING SET AND TESTING SET

```
[ ] from sklearn.model_selection import train_test_split
    x_train,x_test,y_train,y_test=train_test_split(x_scaled,y,test_size=0.2,random_state=0)
```

```
print(x_train.shape,x_test.shape,y_train.shape,y_test.shape)
```

```
(320, 3) (80, 3) (320,) (80,)
```

MODEL BUILDING

Our objective is to try working using various ml algorithms and figuring out the best algorithm suited for this particular use case

1)Logistic Regression

```
[ ] from sklearn.linear_model import LogisticRegression
    lr=LogisticRegression()
```

```
lr.fit(x_train,y_train)
```

```
LogisticRegression()
```

```
[ ] pred=lr.predict(x_test)
    pred
```

```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1])
```

```
[ ] #random value prediction
    lr.predict(ms.transform([[1,19,19000]]))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but MinMaxScaler was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LogisticRegression was fitted with feature names
  warnings.warn(
array([0])
```

EVALUATION OF MODEL

```
[ ] from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,roc_auc_score,roc_curve
```

```
[ ] score1=accuracy_score(y_test,pred)
    score1
```

```
0.925
```

```
[ ] confusion_matrix(y_test,pred)
```

```
array([[58,  0],
       [ 6, 16]])
```

```
pd.crosstab(y_test,pred)
```

col_0	0	1
Purchased		
0	58	0
1	6	16

```
[ ] print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.91      1.00      0.95        58
           1       1.00      0.73      0.84        22

    accuracy                           0.93        80
   macro avg       0.95      0.86      0.90        80
weighted avg       0.93      0.93      0.92        80
```

2) K_nearest neighbors[KNN]

```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier()
knn.fit(x_train,y_train)
```

```
KNeighborsClassifier
KNeighborsClassifier()
```

```
y_pred=knn.predict(x_test)
y_pred
```

```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1])
```

```
#random val prediction
knn.predict([[1,19,19000]])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but KNeighborsClassifier was fitted with feature names
  warnings.warn(
array([1])
```

EVALUATNG THE MODEL

```
[ ] from sklearn.metrics import accuracy_score,classification_report
```

```
[ ] score=accuracy_score(y_pred,y_test)
    score
```

```
0.95
```

```
[ ] print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.98      0.95      0.96        58
           1       0.88      0.95      0.91        22

    accuracy                           0.95        80
   macro avg       0.93      0.95      0.94        80
weighted avg       0.95      0.95      0.95        80
```

```
pd.crosstab(y_test,y_pred)
```

col_0	0	1
Purchased		
0	55	3
1	1	21

3) Decision tree classification

```
[ ] from sklearn.tree import DecisionTreeClassifier
    dtc=DecisionTreeClassifier()
    dtc.fit(x_train,y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

```
    pred2=dtc.predict(x_test)
    pred2
```

```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1])
```

```
[ ] #random value prediction
    dtc.predict(ms.transform([[1,19,19000]]))
```

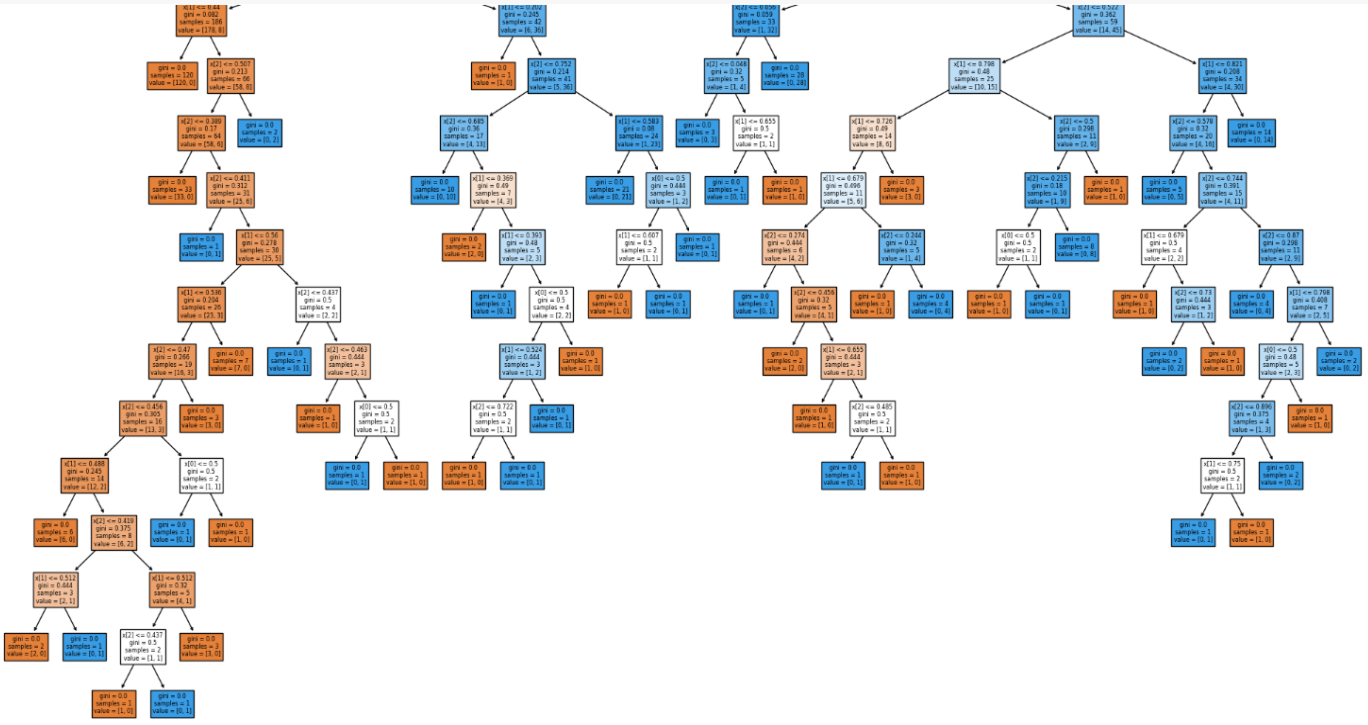
```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but MinMaxScaler was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but DecisionTreeClassifier was fitted with feature names
  warnings.warn(
array([0])
```

```
[ ] score2=accuracy_score(y_test,pred2)
    score2
```

```
0.925
```

HYPER PARAMETER TUNING

```
[ ] from sklearn import tree
    plt.figure(figsize=(25,15))
    tree.plot_tree(dtc,filled=1)
```

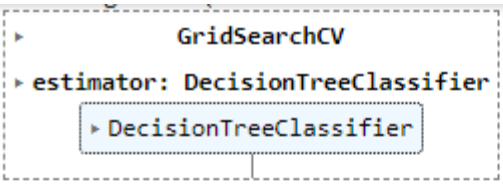


```
[ ] from sklearn.model_selection import GridSearchCV
    parameter={
        'criterion':['gini','entropy'],'splitter':['best','random'],'max_depth':[1,2,3,4,5],'max_features':['auto','sqrt','log2']

    }
```

```
[ ] grid_search=GridSearchCV(estimator=dtc,param_grid=parameter,cv=5,scoring='accuracy')
```

```
[ ] grid_search.fit(x_train,y_train)
```

```
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'`.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'`.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'`.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'`.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'`.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'`.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'`.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'`.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'`.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'`.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'`.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'`.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'`.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'`.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'`.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'`.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'`.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'`.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'`.
  warnings.warn(
```




```
[ ] grid_search.best_params_
```

```
{'criterion': 'entropy',
 'max_depth': 4,
 'max_features': 'log2',
 'splitter': 'best'}
```

```
[ ] dtc_cv=DecisionTreeClassifier(criterion= 'gini',
     max_depth= 4,
     max_features='log2',
     splitter='best')
    dtc_cv.fit(x_train,y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=4, max_features='log2')
```

```
pred_cv=dtc_cv.predict(x_test)
```

```
[ ] print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.91      1.00      0.95        58
           1       1.00      0.73      0.84        22

    accuracy                           0.93        80
   macro avg       0.95      0.86      0.90        80
weighted avg       0.93      0.93      0.92        80
```

```
[ ] score_cv=accuracy_score(y_test,pred_cv)
    score_cv
```

```
0.95
```

4) random forest classification

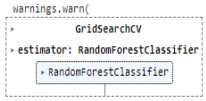
```
[ ] from sklearn.ensemble import RandomForestClassifier
    rfc=RandomForestClassifier()
```

```
[ ] forest_params=[{'max_depth':list(range(10,15)),'max_features':list(range(0,14))}]
```

```
[ ] rfc_cv=GridSearchCV(rfc,param_grid=forest_params,cv=10,scoring='accuracy')
```

```
[ ] rfc_cv.fit(x_train,y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_validation.py:378: FitFailedWarning:
50 fits failed out of a total of 700.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
50 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_validation.py", line 686, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py", line 340, in fit
    self._validate_params()
  File "/usr/local/lib/python3.10/dist-packages/sklearn/base.py", line 600, in _validate_params
    validate_parameter_constraints(
  File "/usr/local/lib/python3.10/dist-packages/sklearn/utils/_param_validation.py", line 97, in validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'max_features' parameter of RandomForestClassifier must be an int in the range [1, inf), a float in the range (0.0, 1.0], a str among {'log2', 'auto' (deprecated), 'sqrt'} or None. Got 0 instead.

  warnings.warn(some_fits_failed_message, FitFailedWarning)
/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_search.py:952: UserWarning: One or more of the test scores are non-finite: [     nan 0.884375 0.8875   0.884375 0.890625 0.878125 0.88125  0.8875
 0.875    0.878125 0.884375 0.875    0.86875  0.878125      nan 0.884375
 0.890625 0.878125 0.878125 0.884375 0.878125 0.878125 0.88125  0.86875
 0.884375 0.884375 0.88125  0.88125       nan 0.884375 0.8875   0.878125
 0.878125 0.878125 0.875    0.878125 0.8875   0.878125 0.8625   0.865625
 0.884375 0.884375      nan 0.8875   0.8875   0.884375 0.865625 0.871875
 0.878125 0.878125 0.875    0.88125  0.884375 0.875    0.890625 0.8875
      nan 0.884375 0.8875   0.884375 0.884375 0.884375 0.878125 0.884375
 0.88125  0.875    0.88125  0.88125  0.88125  0.88125 ]
  warnings.warn(
```




```
[ ] pred3=rfc_cv.predict(x_test)
```

```
[ ] # print(classification_report(y_test,pred))
```

```
score4=accuracy_score(y_test,pred3)
score4
```

```
0.925
```

5) Support Vector Machine (SVM) classification

```
[ ] from sklearn.svm import SVC
    model=SVC(probability=True)
```

```
[ ] rand_list={"C":[2,3,4,5,7,8,10],"gamma":[0.1,0.2,0.3,0.5,0.6,0.8]}#c  values tells us about how the hyperplane is created
    #if c is less then low hyperplane is created.if c is high error is high
    #gamma value tells us how loosly the training data is fitted into the model
```

```
[ ] from sklearn.model_selection import RandomizedSearchCV
    rand_search=RandomizedSearchCV(model,param_distributions=rand_list,n_iter=20,cv=5)
```

```
    rand_search.fit(x_train,y_train)
```

```
RandomizedSearchCV
  estimator: SVC
    SVC
```

```
[ ] pred5=rand_search.predict(x_test)
    pred5
```

```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1])
```

```
[ ] rand_search.predict(ms.transform([[1,19,19000]]))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but MinMaxScaler was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
array([0])
```

```
[ ] score5=accuracy_score(y_test,pred5)
    score5
```

```
0.95
```

6) Naive bayes

```
[ ] from sklearn.naive_bayes import GaussianNB
    nb=GaussianNB()
    nb.fit(x_train,y_train)
```

```
GaussianNB
GaussianNB()
```

```
[ ] pred6=nb.predict(x_test)
    pred6
```

```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1])
```

```
    #random value prediction
    nb.predict(ms.transform([[1,19,19000]]))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but MinMaxScaler was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but GaussianNB was fitted with feature names
  warnings.warn(
array([0])
```

```
[ ] cm6 = confusion_matrix(y_test, pred6)
```

```
[ ] print(cm6)
```

```
[[56  2]
 [ 4 18]]
```

```
[ ] score7=accuracy_score(y_test,pred6)
    score7
```

```
0.925
```

```
[ ] score7=accuracy_score(y_test,pred6)
    score7
```

```
0.925
```

```
[ ] pd.crosstab(y_test,pred)
```

col_0	0	1
Purchased		
0	58	0
1	6	16

```
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.91      1.00      0.95        58
           1       1.00      0.73      0.84        22

    accuracy                           0.93        80
   macro avg       0.95      0.86      0.90        80
weighted avg       0.93      0.93      0.92        80
```

Inference

Among all implemented algorithms , Support Vector Machine(SVC) , KNN and Decision Tree Classifier perform well on Car Purchase Dataset .
Accuracy of KNN,SVM and DTC : 95%